


|   |                       |
|---|-----------------------|
|  <b>MOTION IMAGERY<br/>STANDARDS BOARD</b> | <b>MISB ST 1904.1</b> |
| <b>STANDARD</b>   |                       |
| <b>Motion Imagery Metadata (MIMD): Base Attributes</b>  | <b>25 June 2020</b>   |

## 1 Scope

The Motion Imagery Metadata (MIMD) documents are composed of the MIMD Modeling Rules, MIMD Model, and Model-to-KLV Transmutation Instructions. MISB ST 1903 [1] defines the MIMD Model, which is a UML model organized as a hierarchy of information including metadata for temporal, platform, payload, sensor, command, automated processes, exploitation, security, and more. MISB ST 1901 [2] defines the modeling rules which are a subset of standardized UML class diagrams specifically minimized to enable transforming, or transmuting, class instances to KLV. MISB ST 1902 [3] defines the Model-to-KLV Transmutation Instructions for constructing bandwidth efficient KLV structures from the model.

This standard defines the MIMD Base Attributes, which comprises a common set of attributes supporting all classes across the suite of MIMD standards.

## 2 References

- [1] MISB ST 1903.1 Motion Imagery Metadata (MIMD): Model, Jun 2020.
- [2] MISB ST 1901.1 Motion Imagery Metadata (MIMD): Modeling Rules, Jun 2020.
- [3] MISB ST 1902.1 Motion Imagery Metadata (MIMD): Model-to-KLV Transmutation Instructions, Jun 2020.
- [4] MISB ST 0107.4 KLV Metadata in Motion Imagery, Feb 2019.
- [5] MISB ST 1010.3 Generalized Standard Deviation and Correlation Coefficient Metadata, Oct 2016.

## 3 Revision History

| Revision  | Date      | Summary of Changes  |
|-----------|-----------|---|
| ST 1904.1 | 6/25/2020 | <ul style="list-style-type: none"> <li>• Restructured to enable auto-generation of Model Class and Model Enumeration sections</li> <li>• Changed mimdId from UInt to UInt[≤2]</li> <li>• Added groupId to mimdId</li> </ul> |

## 4 Acronyms, Terms, Definitions

|             |   |
|-------------|---|
| <b>FLP</b>  | Floating Length Pack                            |
| <b>KLK</b>  | Key Length Value                                |
| <b>MIMD</b> | Motion Imagery Metadata                         |
| <b>MISB</b> | Motion Imagery Standards Board                  |
| <b>SDCC</b> | Standard Deviation and Correlation Coefficients |
| <b>UML</b>  | Unified Modeling Language                       |

## 5 Introduction

The MIMD Model contains UML classes with each class having a set of attributes for the modeling item or concept. To enable a set of common features for all classes, each class includes a set of common attributes, called Base Attributes. The features are: ability to link class instances together independent of the class hierarchy, per instance timing information, uncertainty information (i.e., standard deviation and correlations), and per instance security marking.

## 6 Model Classes

### 6.1 Base Class

The MIMD Base class provides a common set of attributes and relationships to all classes in the MIMD Model. Thus, every class in the MIMD Model includes attributes and functionality from this common list of MIMD Base Attributes. To reduce the visual complexity of the UML model, the classes in the MIMD Model do not show the MIMD Base Attributes; however, they are present in the MIMD Model classes which are not inheriting attributes from a generalization class. Classes which inherit attributes from their parent also inherit the base attributes of the parent.

Figure 1 shows the UML for the MIMD Base Attributes. The UML shows a Base “class” diagram; however, consider the “class” as a set of attributes every class includes. The following sub-sections provide details on the MIMD Base Attributes.

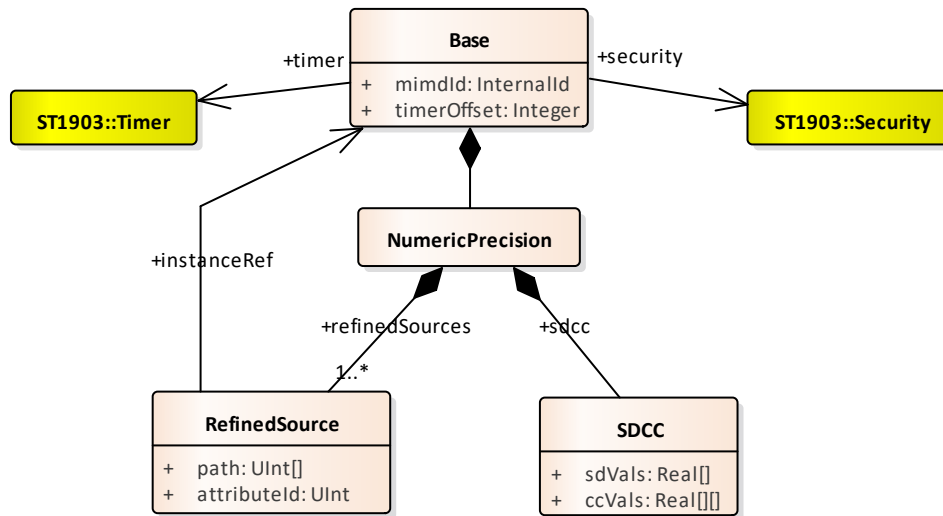
**Figure 1: Base Class UML**

Table 1 lists the MIMD Base Attributes currently in use. The MIMD model currently reserves base attributes 6 through 32 for future use.

**Table 1: Base Class Attributes**

| Id | Name             | Type               | Min | Max | Res | MLen | Units | Ref   |
|----|------------------|--------------------|-----|-----|-----|------|-------|-------|
| 1  | mimdId           | Tuple              | N/A | N/A | N/A | 2    | None  | 6.1.1 |
| 2  | timer            | REF<Timer>         | N/A | N/A | N/A | N/A  | None  | 6.1.2 |
| 3  | timerOffset      | Integer            | --  | --  | N/A | N/A  | ns    | 6.1.3 |
| 4  | numericPrecision | NumericalPrecision | N/A | N/A | N/A | N/A  | None  | 6.1.4 |
| 5  | security         | REF<Security>      | N/A | N/A | N/A | N/A  | None  | 6.1.5 |

### 6.1.1 Attribute 1 – mimdId

The MIMD instance Identifier (mimdId) uniquely identifies a specific instance of a class within the MIMD Model.

The mimdId is a Tuple containing two unsigned integers: serial number and group identifier. The first unsigned integer is an instance identifier serial number. The minimum serial number value is one; however, classes may have attributes that reference a serial number set to zero to indicate special meanings (e.g., a MISB ST 1906 Stage class instance uses a parentStage reference of zero to signify the Stage is a root stage).

The second unsigned integer is the group identifier to which the instance belongs. The minimum group identifier is zero, which is the default group. Transmutations may remove the default group from the mimdId to save bandwidth. Section 6.1.1.1 provides background on the use of group identifiers. Together the serial number and group identifier provide a unique identity within the whole MIMD Model instance hierarchy.

When a class has a unique identity, other classes may reference it outside of the context of the data hierarchy, which creates a link between the two classes – this is the Directed Association relationship discussed in MISB ST 1901.

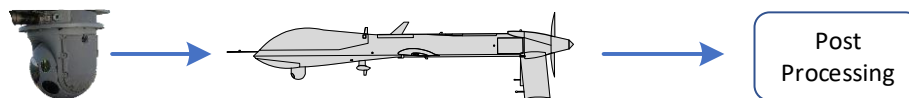
When creating class instances the `mimId` attribute is optional unless another class instance references it. When a class instance does not have other class instances referencing it there is no need to create a `mimId`, thus saving bandwidth.

To allow for packet-to-packet instance correlation (e.g., Report-on-Change), the `mimId` remains constant for a class instance from MIMD Packet-to-MIMD Packet for the duration of the MIMD Stream.

| Requirement(s) |  |
|----------------|--|
| ST 1904-01     | Where a class instance defines a MIMD Identifier ( <code>mimId</code> ), the <code>mimId</code> shall be unique among all MIMD instance Identifiers in the MIMD Stream.      |
| ST 1904-02     | Where a class instance is used in multiple MIMD Packets, the MIMD Identifier ( <code>mimId</code> ) shall use the same value.  |
| ST 1904-03     | When a class instance is referenced by another MIMD class instance, the referenced instance's MIMD Identifier ( <code>mimId</code> ) shall not be <code>ROC_Unknown</code> . |

#### 6.1.1.1 Using Group Identifiers

Group identifiers provide a method to deconflict the serial number address space for two or more systems generating metadata. Motion Imagery Systems are usually collections of sub-systems, configured either in a pipeline, collection, or both. For example, Figure 2 illustrates a sub-system pipeline consisting of a sensor ball passing data onto a platform which is passing data on to some post processing (e.g., VMTI).

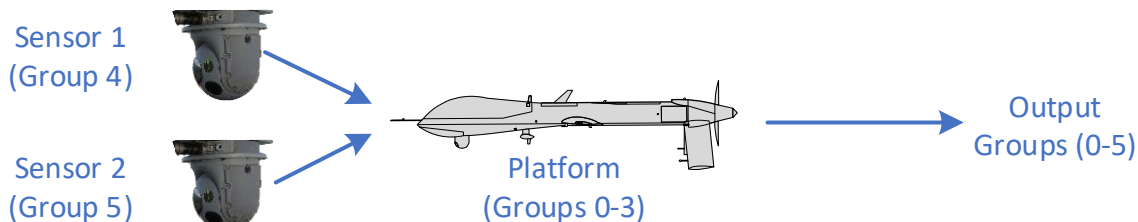


**Figure 2: Example System Pipeline**

For this example, the sensor is upstream of the platform and the platform is upstream of the post processing. Each sub-system is an independently manufactured sub-system, without a process of system level coordination for the serial number address space. Since each sub-system, at any time, can grow their serial number address space as needed, the ability to pre-define the size of the address space is not feasible.

The group identifier enables each sub-system to allocate serial number addresses into one or more groups, therefore each group identifier defines a serial number address space. With each sub-system using different group identifiers, system pipelines can merge independent streams of MIMD data without conflicts of serial number address space. Figure 3 illustrates a system with two sensors (Sensor 1 and Sensor 2) on board a Platform. Each of these sensors is independent and does not communicate with each other to deconflict their serial number address space; therefore, they have each been allocated a group identifier. The Sensors have set the group for their serial number address space as follows: Sensor 1 uses Group 4 and Sensor 2 uses Group 5. The Platform has set multiple groups from Group 0 to Group 3 for its internal use. One of the tasks the platform system performs is to merge the metadata from the different sub-systems (e.g.,

Sensor 1 and Sensor 2) into one composite stream of metadata. With the group identifiers the merging process does not need to deconflict serial numbers.



**Figure 3: Group Identifier Example**

The overall system must coordinate the allocation of sub-system group identifiers, e.g., accomplished via command and control or discovery process. The group allocation methodology is beyond the scope of this standard and left to the implementing system.

As the example illustrates, there are two types of sub-system components: data generators and data integrators. Data generators only create MIMD data using one or more groups (e.g., Sensor 1 and 2 in Figure 3), and pass their MIMD data downstream to data integrators. Data integrators accept one or more upstream (in the pipeline) sub-system's MIMD data (e.g., Platform in Figure 3), potentially add additional MIMD data, and produce a culmination of the MIMD data for other downstream data integrators and eventually consumers.

When a system uses more than one group, data generators and integrators must declare the minimum and maximum group identifiers they allocated (or reuse) in the MIMD class (see MISB ST 1903) using the `groupIdRange` attribute.

In receiving upstream MIMD data the `groupIdRange` attribute provides the necessary information to validate group allocation and merge the metadata. The resulting metadata output of a data integrator will have the `groupIdRange` attribute set.

A sub-system component (either data generator or integrator) sets the maximum number of groups it plans to use. For example, a sensor ball with two sensors has only one active. An hour later the operator activates the second sensor and the second sensor produces metadata with a new group identifier. In this example the sub-system needs to have its `groupIdRange` set to accommodate both groups.

| Requirement(s) |  |
|----------------|--|
| ST 1904.1-11   | Where a Motion Imagery System uses a <code>mimdId</code> group identifier larger than zero, the system shall set the MIMD class <code>groupIdRange</code> attribute (defined in MISB ST 1903) to encompass all the group identifiers in use. |
| ST 1904.1-12   | The MIMD class <code>groupIdRange</code> attribute's maximum group identifier (defined in MISB ST 1903) shall not decrease over time.  |

### 6.1.2 Attribute 2 – timer

*This attribute is a directed association (or reference) to Class Timer. MISB ST 1903 defines the Timer class.*

The MIMD class contains a list of one or more Timer class instances, see *timers* attribute MISB ST 1903. Each Timer includes a time value in nanoseconds and each timer class instance includes a *mimId*. The *timer* attribute allows any class to associate its attributes to the time value in any of the Timer instances.

When an instance does not set the timer attribute, the parent's timer provides the timing association as the default. If the parent does not set its timer attribute, its parent defines the default, etc. Ultimately, the top-most-class instance, the MIMD class instance, becomes the default. If the MIMD class instance does not define a timer then the first timer in the Timer list becomes the default.

For example, if two sensors use two different timer instances (e.g., GPS Timer and internal clock Timer - each with their own precisions and accuracies), each Sensor instance would set their timer attribute to soft link to the appropriate Timer instance. Each Sensor's measurements (i.e., attributes) are then time tagged to the appropriate timer, enabling receivers to properly correct for time differences between the GPS and internal clock, if needed.

| Requirement(s) |   |
|----------------|---|
| ST 1904-04     | Where a class instance timer attribute is Unknown, the timer's value shall default to the same value as the parent's timer.                                       |
| ST 1904-05     | Where the MIMD class instance timer attribute is Unknown, the MIMD's instance timer value shall default to the <i>mimId</i> of the first Timer in the Timer list. |

### 6.1.3 Attribute 3 – timerOffset

The *timerOffset* is a positive, zero, or negative offset in nanoseconds from the instance's timer value; Section 6.1.2 defines the instance's timer via the timer attribute. The *timerOffset* provides additional measurement accuracy for an instance's attribute values.

For example, if two sensors (Sensor 1 and Sensor 2) use the same time source (e.g., GPS Timer), but Sensor 2's attitude measurements are temporally delayed by 300 milliseconds, Sensor 2's instance sets the *timerOffset*=300,000,000. Sensor 2's measurements can now be correlated temporally with Sensor 1 given the known offset. This enables receivers to properly correct for time differences between the two sensors for exploitation or computation, if needed.

The default value for *timerOffset* is zero. If the value is "Unknown" (see MISB ST 0107 [4] for Report-on-Change), receivers use a *timerOffset* value of zero.

| Requirement |   |
|-------------|---|
| ST 1904-06  | Where a class instance <i>timerOffset</i> attribute is Unknown, the <i>timerOffset</i> attribute value shall default to zero. |

### 6.1.4 Attribute 4 – numericPrecision

*This attribute is a composite relationship to the Class NumericalPrecision. Section 6.2 defines the NumericalPrecision class.*

### 6.1.5 Attribute 5 – security

*This attribute is a directed association (or reference) to Class Security. MISB ST 1903 defines the Security class.*

The MIMD class contains a list of one or more Security class instances, see *securityOptions* attribute in MISB ST 1903. Each Security class instance includes classification information and a *mimId*. The security attribute allows any model instance to associate its attributes to the classification information in any of the Security class instances.

When a model instance does not set the security attribute, the parent instance provides the association as the default. If the parent does not set its security attribute, its parent defines the default, etc. Ultimately, the top-most-class instance, the MIMD class, becomes the default. If the MIMD class does not define the security attribute, then the first Security class instance in the Security class list becomes the default.

| Requirement(s) |  |
|----------------|--|
| ST 1904-09     | Where a class instance security attribute is Unknown, the security value shall default to the same value as the parent's security.   |
| ST 1904-10     | Where the MIMD class instance security attribute is Unknown, the MIMD's instance security value shall default to the <i>mimId</i> of the first Security instance in the Security list. |

## 6.2 Numerical Precision Class

Numerical Precision metadata aide analyzing measurement values and includes standard deviation and correlation coefficient information. Standard deviation is a quantity of variation (or error) of a random variable's measurements. In this context a random variable is any numeric metadata. A correlation coefficient is a measure of the degree two random variables affect one another. A correlation coefficient value ranges from -1 to 1, with -1 meaning the two random variables are anti-correlated, zero meaning they are independent, and 1 meaning they are perfectly correlated to each other.

MISB ST 1010 [5] defines an efficient method – the SDCC-FLP – constructed as a floating length pack (FLP) for compacting standard deviation and correlation coefficient information metadata using KLV. Together the standard deviation and correlation coefficient information form a two-dimensional, square matrix of values called an SDCC Matrix. MISB ST 1010 relies on first defining a *Full Source List* of Local Set items from which a *Refined Source List* precedes an SDCC-FLP to specify the random variables that represent the rows/columns in the SDCC Matrix.

This method, however, fails to work directly in the MIMD Model because: 1) the MIMD Model is a data hierarchy; 2) the MIMD Model has no mechanism to specify an ordering of attributes; and 3) the method is optimized for KLV and not applicable to other MIMD Formats. To define the equivalent of a Refined Source List in the MIMD Model, a list of RefinedSource classes defines a link between each hierarchical attribute to a row/column of the SDCC Matrix.

The *Full Source List* for the MIMD Model is composed of all the numerical attributes in the hierarchy of classes. The *Refined Source List* for the MIMD Model is a class's list of RefinedSource instances.

Table 2 identifies the attributes in the StdDevAndCorrelation class.

**Table 2: NumericalPrecision Class Attributes**

| Id | Name           | Type                | Min | Max | Res | MLen | Units | Ref   |
|----|----------------|---------------------|-----|-----|-----|------|-------|-------|
| 33 | refinedSources | LIST<RefinedSource> | 1   | *   | N/A | N/A  | None  | 6.2.1 |
| 34 | sdcc           | SDCC                | N/A | N/A | N/A | N/A  | None  | 6.2.2 |

### 6.2.1 Attribute 33 – refinedSources

*This attribute is a list of RefinedSource classes. Section 6.3 defines the RefinedSource class.*

### 6.2.2 Attribute 34 – sdcc

*This attribute is a composite relationship to the Class SDCC. Section 6.4 defines the SDCC class.*

## 6.3 RefinedSource Class

The RefinedSource class contains attributes for connecting or “linking” an attribute in the model to a row/column in the SDCC matrix. This class uses the metadata within the model itself to create the connection, specifically the mimdId and attribute identifiers.

There are several necessary criteria when connecting class attributes to the rows/columns of the SDCC Matrix:

- The naming method supports linking any children attributes in the hierarchy, at or below, the current class instance (e.g., attributes of a child class of a child class) to the SDCC Matrix
- The naming method supports linking any individual Composition relationship “list” element’s child attributes to the SDCC Matrix
- The naming method supports linking attributes from classes in an inheritance relationship
- The naming method supports ordering the instance attributes in any order that defines the order in the rows/columns of the SDCC Matrix
- The naming method supports Report-on-Change. This means the naming method needs to support class attributes that may be Known-Dynamic and Known-Static.
- The naming method supports the possibility of Known-Static or Known-Dynamic values changing to Unknown

To illustrate how a hierarchy of classes and attributes maps to an SDCC Matrix, Figure 4 shows an example class model, while Figure 5 shows an example instance of the model (denoted by the underlined class names). For this example, the metadata producer is adding standard deviations and correlation coefficients to the Sensor class which include attributes of the Sensor and its children classes, where the children classes also have inheritance relationships. The directed links (arrows) in Figure 5 represent the inheritance of those instances.



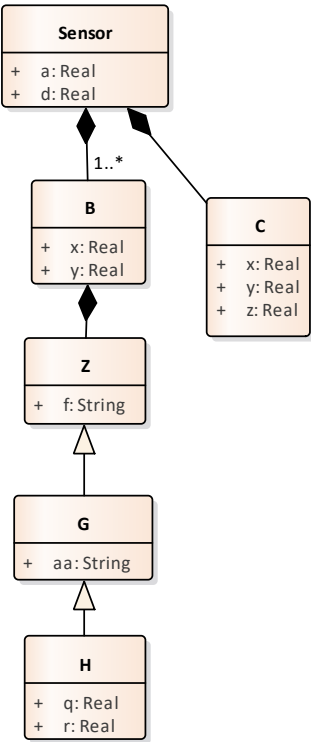


Figure 4: Example Class Model

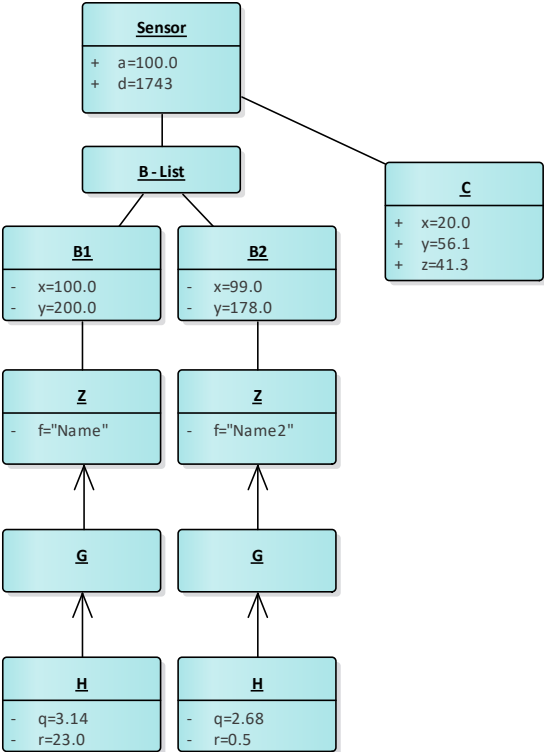


Figure 5: Example Instance of the Model

Table 3 shows the same information as Figure 5 expressed in outline format; the red items identify those included in the SDCC matrix. Numbers on the list elements indicate their index in a list (e.g., B1 is the first element in the list). The subscripted numbers on the attributes (e.g., H<sub>1</sub>) identifies individual instances of the same class. The items in red are the items linking to rows/columns in the SDCC Matrix (shown on the right). The “dot” notation, (e.g., B1.x) indicates which instance and attribute is in the SDCC Matrix. The brace notation “{ }” indicates the attribute represents an instance with an inheritance relationship to its parent. This example shows all the possible types of “links” necessary from an inheritance hierarchy to an SDCC matrix.

**Table 3: Instances to SDCC Matrix Naming Example**

| Example Instance  |      |      |   |     |      |      |     |      |      |
|-------------------|------|------|---|-----|------|------|-----|------|------|
| Sensor Attributes |      |      |   |     |      |      |     |      |      |
| a                 |      |      |   |     |      |      |     |      |      |
| B – (List)        |      |      |   |     |      |      |     |      |      |
| B1                |      |      |   |     |      |      |     |      |      |
| x                 |      |      |   |     |      |      |     |      |      |
| y                 |      |      |   |     |      |      |     |      |      |
| Z <sub>1</sub>    |      |      |   |     |      |      |     |      |      |
| f                 |      |      |   |     |      |      |     |      |      |
| {G}               |      |      |   |     |      |      |     |      |      |
| {H <sub>1</sub> } |      |      |   |     |      |      |     |      |      |
| q                 |      |      |   |     |      |      |     |      |      |
| r                 |      |      |   |     |      |      |     |      |      |
| B2                |      |      |   |     |      |      |     |      |      |
| x                 |      |      |   |     |      |      |     |      |      |
| y                 |      |      |   |     |      |      |     |      |      |
| Z <sub>2</sub>    |      |      |   |     |      |      |     |      |      |
| f                 |      |      |   |     |      |      |     |      |      |
| {G}               |      |      |   |     |      |      |     |      |      |
| {H <sub>2</sub> } |      |      |   |     |      |      |     |      |      |
| q                 |      |      |   |     |      |      |     |      |      |
| r                 |      |      |   |     |      |      |     |      |      |
| C                 |      |      |   |     |      |      |     |      |      |
| x                 |      |      |   |     |      |      |     |      |      |
| y                 |      |      |   |     |      |      |     |      |      |
| z                 |      |      |   |     |      |      |     |      |      |
| d                 |      |      |   |     |      |      |     |      |      |
| SDCC Matrix       |      |      |   |     |      |      |     |      |      |
|                   | B1.x | B1.y | d | C.z | B2.x | B2.y | C.y | H1.q | H2.q |
| B1.x              |      |      |   |     |      |      |     |      |      |
| B1.y              |      |      |   |     |      |      |     |      |      |
| d                 |      |      |   |     |      |      |     |      |      |
| C.z               |      |      |   |     |      |      |     |      |      |
| B2.x              |      |      |   |     |      |      |     |      |      |
| B2.y              |      |      |   |     |      |      |     |      |      |
| C.y               |      |      |   |     |      |      |     |      |      |
| H1.q              |      |      |   |     |      |      |     |      |      |
| H2.q              |      |      |   |     |      |      |     |      |      |

This example shows the naming method needs to:

- Link Sensor Attributes (e.g., d) to the SDCC Matrix
- Link child class attributes (e.g., C.y and C.z) to the SDCC Matrix
- Link list element attributes (e.g., B1.x, B1.y) to the SDCC Matrix
- Link inheritance child attributes (e.g., H1.q, H2.q) to the SDCC Matrix

Additionally, the example illustrates the arbitrary ordering of the elements in the hierarchical Sensor Attributes instance versus the fixed row/column order of the SDCC Matrix. The order of hierarchical attribute metadata may be different in every packet. The row/column order of the SDCC Matrix is usually the same for a period. The Refined Source List maps the hierarchical attribute to a row/column in the SDCC Matrix in an ordered list. Once choosing a selected order the SDCC Matrix needs to retain that same order.

To provide a clear indication of which instances and which attributes link to the SDCC Matrix, the naming method employs the *mimId* and attribute identifiers. All class instances linking to an SDCC Matrix need to reference or create a *mimId*.

| Requirement |   |
|-------------|---|
| ST 1904-07  | Where a class instance attributes are linked to an SDCC Matrix and the class instance is a child of an inheritance relationship, the class' top-most generalization parent shall define a mimdId. |

Children classes in an inheritance relationship will not have their own mimdId; they inherit their mimdId from their parent. If the parent is also a child of an inheritance relationship, it inherits its mimdId from its parent, and so on, to the top-most generalization class.

To provide the linkage between an instance attribute and the SDCC Matrix rows/columns, the Refined Source class has three attributes: **mimdId reference**, **path**, and **attribute identifier**. The **mimdId reference** is the mimdId for the class instance providing the attribute to link to a row/column in the SDCC Matrix. A **path** is a list of attribute identifiers describing how to descend the model hierarchy to the final attribute identifier when the attribute identifier is in an inheriting class. Where the attributes are not a part of an inheriting class there is no need for a path. The **attribute identifier** is the attribute to link to a row/column in the SDCC Matrix.

The order of the SDCC Matrix rows/columns needs to match the order of the RefinedSource instances in the Refined Source List. For example, Table 4 extends the Table 3 example with the inclusion of a mimdId for each class and includes attribute identifiers in parenthesis after each attribute. The middle column, Refined Source List (Textual Names), is a list of class instances to attribute names using the textual names. The right column shows the same list (i.e. Refined Source List (Numeric)) as the middle column but uses the numerical values for the mimdId reference (first value), path (values within brackets “[ ]”), and attribute identifier (last value) for each attribute.

**Table 4: Refined Source List Example**

| Example Instance       | Refined Source List<br>(Textual Names) | Refined Source List<br>(Numeric) |
|------------------------|--|----------------------------------|
| Sensor Attributes      | (B1, x)                                | (9, [], 33)                      |
| mimdlId=7              | (B1, y)                                | (9, [], 34)                      |
| a (33)                 | (Sensor Attributes, d)                 | (7, [], 36)                      |
| B (34) – (List)        | (C, z)                                 | (13, [], 43)                     |
| B1                     | (B2, x)                                | (10, [], 33)                     |
| mimdlId=9              | (B2, y)                                | (10, [], 34)                     |
| <b>x (33)</b>          | (C, y)                                 | (13, [], 42)                     |
| <b>y (34)</b>          | (Z, G, H <sub>1</sub> , q)             | (14, [34, 33], 33)               |
| Z (35)                 | (Z, G, H <sub>2</sub> , q)             | (15, [34, 33], 33)               |
| mimdlId=14             |  |                                  |
| f (33)                 |  |                                  |
| {G} (34)               |  |                                  |
| {H <sub>1</sub> } (33) |  |                                  |
| <b>q (33)</b>          |  |                                  |
| r (34)                 |  |                                  |
| B2                     |  |                                  |
| mimdlId=10             |  |                                  |
| <b>x (33)</b>          |  |                                  |
| <b>y (34)</b>          |  |                                  |
| Z (35)                 |  |                                  |
| mimdlId=15             |  |                                  |
| f (33)                 |  |                                  |
| {G} (34)               |  |                                  |
| {H <sub>2</sub> } (33) |  |                                  |
| <b>q (33)</b>          |  |                                  |
| r (34)                 |  |                                  |
| C (35)                 |  |                                  |
| mimdlId=13             |  |                                  |
| x (41)                 |  |                                  |
| <b>y (42)</b>          |  |                                  |
| <b>z (43)</b>          |  |                                  |
| <b>d (36)</b>          |  |                                  |

The position of a RefinedSource in the list determines which row/column in the SDCC Matrix the attribute is associated. The number of element paths in the list needs to match the number of rows/columns in the SDCC Matrix. For example, in Table 4 the Refined Source List defines nine names (RefinedSources) so the SDCC Matrix is a square matrix with nine rows by nine columns.

| Requirement |   |
|-------------|---|
| ST 1904-08  | The number of RefinedSources in a StdDevAndCorrelation class shall equal the number of rows of its SDCC Matrix. |

Nominally, a RefinedSource attribute is static but may change infrequently. Therefore, it should not consume excessive bandwidth over time when employing the Report-on-Change methodology. For instance, when data attributes linked to an SDCC Matrix become Unknown, none of the SDCC attributes need to change. In this case, the receiver may ignore rows in the SDCC Matrix when processing by setting the standard deviation to zero (see MISB ST 1010).

Table 5 lists the attributes of the RefinedSource class.

**Table 5: RefinedSource Class Attributes**

| Id | Name        | Type      | Min | Max | Res | MLen | Units | Ref   |
|----|-------------|-----------|-----|-----|-----|------|-------|-------|
| 33 | instanceRef | REF<Base> | N/A | N/A | N/A | N/A  | None  | 6.3.1 |
| 34 | path        | UInt[]    | 1   | --  | N/A | N/A  | None  | 6.3.2 |
| 35 | attributeId | UInt      | 1   | --  | N/A | N/A  | None  | 6.3.3 |

### 6.3.1 Attribute 33 – instanceRef

*This attribute is a directed association (or reference) to Class Base. Section 6.1 defines the Base class.*

The *instanceRef* attribute is the identifier (i.e., *mimId*) for the class instance providing the attribute to link to a row/column in the SDCC Matrix. The type “Base\_Ref” is a name of convenience for the type because Base is not a class by itself; it is a list of attributes every class includes.

### 6.3.2 Attribute 34 – path

The *path* attribute is a list of attribute identifiers describing how to descend the model hierarchy to the final attribute identifier. Paths are necessary only when the attribute identifier is in an inheriting class; otherwise there is no need for a path.

### 6.3.3 Attribute 35 – attributeId

The *attributeId* is the attribute to link to a row/column of the SDCC Matrix.

## 6.4 SDCC Class

The SDCC class contains two attributes: Standard Deviations (*sdVals*), and Cross Correlations (*ccVals*). The *sdVals* and *ccVals* attributes together form the SDCC Matrix, a symmetric matrix as described in MISB ST 1010.

The dimensionality and numerical ranges (min/max) of the *sdVals* and *ccVals* values are different, so each have their own variables. In the SDCC Matrix the *sdVals* is the diagonal of the matrix and therefore is a one-dimension array. The numerical ranges for *sdVals* values are specific to the random variables or measurements they reference.

The *ccVals* are the upper triangular portion of the SDCC matrix and therefore a two-dimensional array. The upper triangular matrix has one less row and column than the SDCC Matrix; for

example, if there are 10 variables in the *sdVals* array, the *ccVals* will be a 9x9 with only the upper triangular values defined. The numerical range for all the *ccVals* is from -1.0 to +1.0.

Decoupling the two attribute arrays allows independent use of only one; for example, an application may specify standard deviation without cross correlations.

Table 6 lists the class attributes for the SDCC class.

**Table 6: SDCC Class Attributes<sup>1</sup>**

| <b>Id</b> | <b>Name</b>   | <b>Type</b> | <b>Min</b> | <b>Max</b> | <b>Res</b> | <b>MLen</b> | <b>Units</b> | <b>Ref</b> |
|-----------|---------------|-------------|------------|------------|------------|-------------|--------------|------------|
| 33        | <i>sdVals</i> | Real[]      | 0.0        | --         | --         | N/A         | None         | 6.4.1      |
| 34        | <i>ccVals</i> | Real[][]    | -1.0       | 1.0        | --         | N/A         | None         | 6.4.2      |

### 6.4.1 Attribute 33 – *sdVals*

The *sdVals* attribute is an array of values with its length matching the number of elements in the Refined Source List. The Units for each *sdVals* attribute value match the units of the referred attribute value.

### 6.4.2 Attribute 34 – *ccVals*

The *ccVals* attribute is a two-dimensional square array of values with each dimension matching the number of elements in the Refined Source List. Only the upper triangular values have significance and all other values should be set to zero. Each *ccVals* attribute is unitless and ranges from -1.0 to 1.0, with -1.0 indicating two random variables are anti-correlated, zero meaning they are independent, and +1.0 meaning they are perfectly correlated to each other. Inserting the *sdVals* as the diagonal of the *ccVals* matrix forms the whole SDCC Matrix as defined in MISB ST 1010.

## 7 Model Enumerations

This document does not define enumerations.

---

<sup>1</sup> For consistency across the MIMD suite of documents, MISB ST 1901 defines the label designations for the columns in the attribute tables.